HDFS 负载均衡源码分析

(高级分布式系统 课程研究报告)

21214801 吴坎

中山大学 计算机学院

二〇二二年六月

HDFS：Hadoop 分布式文件系统

- 在廉价的设备上提供高容错、高吞吐量的存储能力

HDFS 很容易出现节点间磁盘利用率不平衡的情况

- 集群新增、删除节点
- 某个机器存储达到饱和值

后果：Map 任务可能会被分配没有存储数据的机器

- 降低本地计算效率
- 浪费网络带宽

结论：需要负载均衡

- 对各节点数据的存储分布进行调整
- 让数据均匀分布，均衡 IO 性能，防止热点发生

实现负载均衡需要满足的原则

- 数据平衡不能导致数据块减少，数据块备份丢失
- 管理员可以中止数据平衡进程
- 每次移动的数据量以及占用的网络资源，必须是可控的
- 数据均衡过程，不能影响 NameNode 的正常工作

负载均衡原理与流程

- NameNode 生成 DataNode 数据分布与磁盘使用情况报告
- 汇总待移动的数据分布情况，计算具体数据块迁移路线图
- 开始数据块迁移，PSDN（Proxy Source Data Node）复制一块待移动数据块

负载均衡原理与流程（续）

- 将复制的数据块复制到目标 DataNode 上并删除原始数据块
- 目标 DataNode 向 PSDN 确认该数据块迁移完成
- PSDN 向 Rebalancing Server 确认本次数据块迁移完成
- 继续执行这个过程，直至集群达到数据均衡标准

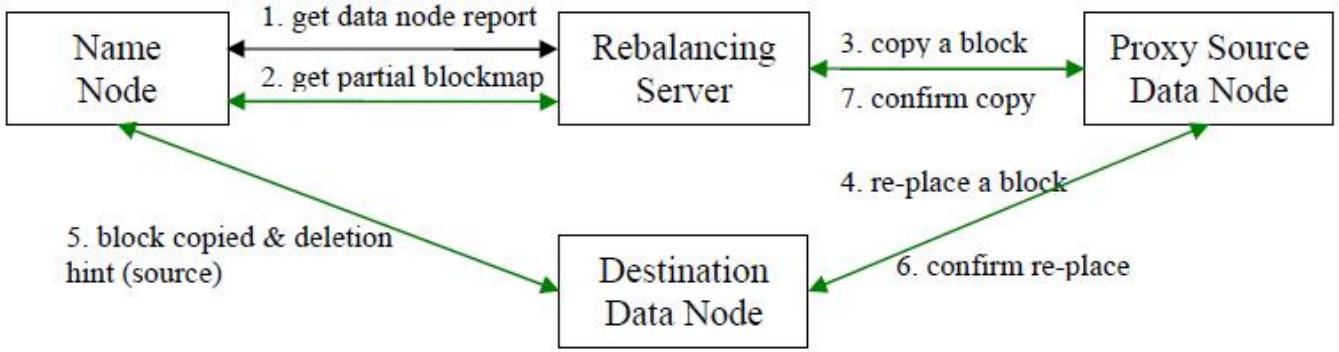

Figure 1: 负载均衡原理图

数据块分组与迁移策略

- 根据阈值划分到 Over、Above、Below、Under 四个组
- 将 Over 组、Above 组中的块向 Below 组、Under 组移动

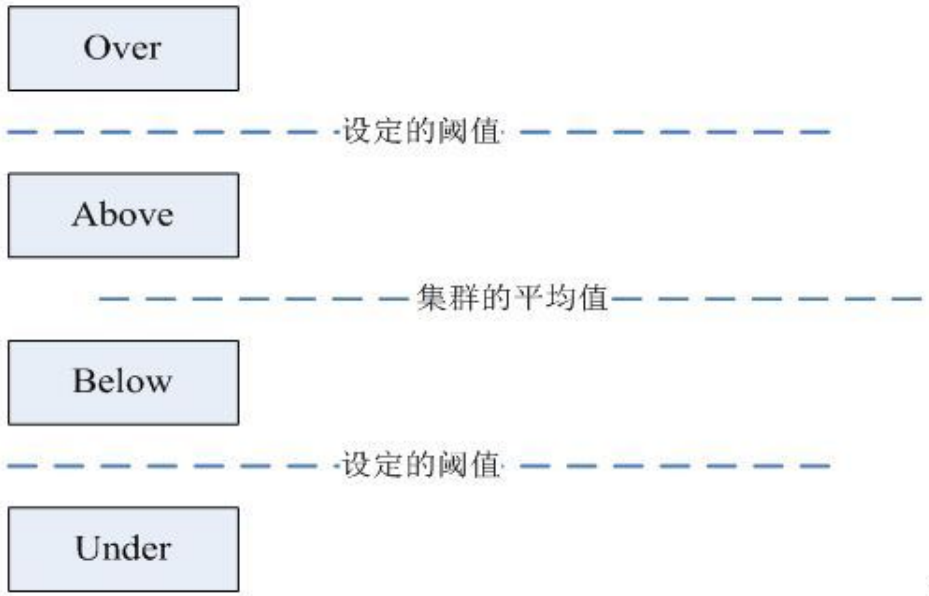

Figure 2: 分组策略

调用链

- Balancer.main → run → doBalance → runOneIteration

```
721     /**
722      * Balance all namenodes.
723      * For each iteration,
724      * for each namenode,
725      * execute a {@link Balancer} to work through all datanodes once.
726      */
727     static private int doBalance(Collection<URI> namenodes,
728         Collection<String> nsIds, final BalancerParameters p, Configuration conf)
729         throws IOException, InterruptedException {
```

Figure 3: https://github.com/apache/hadoop/blob/release-3.3.3-RC1/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/balancer/Balancer.java#L727

```java
/** Run an iteration for all datanodes. */
Result runOneIteration() {
  try {
    final List<DatanodeStorageReport> reports = dispatcher.init();
    final long bytesLeftToMove = init(reports);
    if (bytesLeftToMove == 0) {
      return newResult(ExitStatus.SUCCESS, bytesLeftToMove, 0);
    } else {
      LOG.info( "Need to move "+ StringUtils.byteDesc(bytesLeftToMove)
          + " to make the cluster balanced." );
    }

    // Should not run the balancer during an unfinalized upgrade, since moved
    // blocks are not deleted on the source datanode.
    if (!runDuringUpgrade && nnc.isUpgrading()) {
      System.err.println("Balancer exiting as upgrade is not finalized, "
          + "please finalize the HDFS upgrade before running the balancer.");
      LOG.error("Balancer exiting as upgrade is not finalized, "
          + "please finalize the HDFS upgrade before running the balancer.");
      return newResult(ExitStatus.UNFINALIZED_UPGRADE, bytesLeftToMove, -1);
    }

    /* Decide all the nodes that will participate in the block move and
     * the number of bytes that need to be moved from one node to another
     * in this iteration. Maximum bytes to be moved per node is
     * Min(1 Band worth of bytes,  MAX_SIZE_TO_MOVE).
     */
    final long bytesBeingMoved = chooseStorageGroups();
    if (bytesBeingMoved == 0) {
      System.out.println("No block can be moved. Exiting...");
      return newResult(ExitStatus.NO_MOVE_BLOCK, bytesLeftToMove, bytesBeingMoved);
    } else {
      LOG.info("Will move {}  in this iteration for {}",
          StringUtils.byteDesc(bytesBeingMoved), nnc.toString());
      LOG.info("Total target DataNodes in this iteration: {}",
          dispatcher.moveTasksTotal());
    }

    /* For each pair of <source, target>, start a thread that repeatedly
     * decide a block to be moved and its proxy source,
     * then initiates the move until all bytes are moved or no more block
     * available to move.
     * Exit no byte has been moved for 5 consecutive iterations.
     */
    if (!dispatcher.dispatchAndCheckContinue()) {
      return newResult(ExitStatus.NO_MOVE_PROGRESS, bytesLeftToMove, bytesBeingMoved);
    }

    return newResult(ExitStatus.IN_PROGRESS, bytesLeftToMove, bytesBeingMoved);
  } catch (IllegalArgumentException e) {
```

Figure 4: https://github.com/apache/hadoop/blob/release-3.3.3-RC1/hadoop-hdfs-project/hadoop-hdfs/src/main/java/org/apache/hadoop/hdfs/server/balancer/Balancer.java#L659

负载均衡过程的核心是一个数据均衡算法

- 不断迭代数据均衡逻辑
- 直至集群内数据均衡为止
- 合理选取参数与均衡时机可以有效提升系统性能

缺点与可能的改进方向

- 阈值需要手动指定
- 阈值越小，理论上负载越均衡，但开销也会变大
- …设计自适应的数据均衡算法，避免突发情况

Questions?

Thank you!